IN THE CLAIMS

Please amend claims 1, 2, 8, 10, 11, 14, 15, 16, 17, 18, 24, 26, 27, 30, 31, 32, 38, 40, 41 and 44 as follows:

1. (CURRENTLY AMENDED) A client-server relational database system, comprising:

a client computer;

a server computer; and

a network connecting the client computer and the server computer;

wherein data from the client computer is encrypted by the client computer and hosted by the server computer, the encrypted data is operated upon by the server computer to produce an encrypted intermediate results set, the encrypted intermediate results set is sent from the server computer to the client computer where [[it]] the encrypted intermediate results set is decrypted and operated upon by the client computer to create an updated intermediate results set, and the updated intermediate results set is then re-encrypted and returned to the server computer where [[it]] the re-encrypted intermediate results set is further operated upon by the server computer ~~before being~~ to generate an encrypted new intermediate results set, which is sent ~~again~~ from the server computer to the client computer to be decrypted and operated upon by the client computer in order to produce actual results for presentation to a user.

2. (CURRENTLY AMENDED) The system of claim 1, wherein the client computer decrypts the intermediate results set, performs one or more operations on the decrypted intermediate results set to generate [[an]] the updated intermediate results set, re-encrypts the updated intermediate results set, and returns the re-encrypted intermediate results set to the server computer.

3. (ORIGINAL) The system of claim 2, wherein the operations comprise logical comparison operations.

4. (ORIGINAL) The system of claim 2, wherein the operations comprise filtering operations.

5. (ORIGINAL) The system of claim 2, wherein the operations comprise sorting operations.

6. (ORIGINAL) The system of claim 1, wherein the server computer executes a round-trip filtering operator that specifies when the intermediate results set is sent from the server computer to the client computer to be operated upon by the client computer and then returned to the server computer to be operated upon by the server computer.

7. (ORIGINAL) The system of claim 6, wherein the round-trip filtering operator sends maybe tuples from the server computer to the client computer, the client computer filters out certain tuples from the maybe tuples, and the server computer receives back only certain tuples from the client computer.

8. (CURRENTLY AMENDED) The system of claim 6, wherein the server receives a query from the client computer, generates a plurality of query execution plans having different placements of the round-trip filtering operator, and chooses one of query execution plans that ~~optimizes~~ comprises an optimal placement of the round-trip filtering operator in a query tree.

9. (ORIGINAL) The system of claim 6, wherein the server executes a last-trip decryption operator that specifies when the intermediate results set is sent from the server computer to the client computer in order to produce actual results.

10. (CURRENTLY AMENDED) The system of claim 9, wherein the server computer receives a query from the client computer, generates a plurality of query execution plans having different placements of the last-trip decryption operator, and chooses one of the query execution plans that ~~optimizes~~ comprises an optimal placement of the last-trip decryption operator in a query tree.

11. (CURRENTLY AMENDED) The system of claim 10, wherein the server computer chooses one of the query execution plans that ~~optimizes~~ comprises an optimal placement for the round-trip filtering operators in the query tree and then chooses one of the query execution plans

that ~~optimizes~~ comprises an optimal placement for the last-trip decryption operator in the query tree.

12. (ORIGINAL) The system of claim 9, wherein the last-trip decryption operator can be pulled up above any unary and binary operator in a query tree, except for a GroupBy operator.

13. (ORIGINAL) The system of claim 12, wherein the GroupBy operator can be pulled above a unary operator other than another GroupBy operator if and only if all columns used in the unary operator are functionally computed by grouping columns of an input relation.

14. (CURRENTLY AMENDED) The system of claim 13, wherein the GroupBy operator can be pulled above a binary operator if:

[[(1)]] a left-side relation of the GroupBy operator has a key; and

[[(2)]] a predicate of the GroupBy operator does not use a result of the GroupBy operator.

15. (CURRENTLY AMENDED) A client-server relational database system, comprising:

a client computer connected to a server computer, wherein data from the client computer is encrypted by the client computer and hosted by the server computer, the encrypted data is operated upon by the server computer to produce an encrypted intermediate results set, the encrypted intermediate results set is sent from the server computer to the client computer where [[it]] the encrypted intermediate results set is decrypted and operated upon by the client computer to create an updated intermediate results set, and the updated intermediate results set is then re-encrypted and returned to the server computer where [[it]] the re-encrypted intermediate results set is further operated upon by the server computer ~~before being~~ to generate an encrypted new intermediate results set, which is sent ~~again~~ from the server computer to the client computer to be decrypted and operated upon by the client computer in order to produce actual results for presentation to a user.

16. (CURRENTLY AMENDED) A client-server relational database system, comprising:

a server computer connected to a client computer, wherein data from the client computer is encrypted by the client computer and hosted by the server computer, the encrypted data is

operated upon by the server computer to produce an encrypted intermediate results set, the encrypted intermediate results set is sent from the server computer to the client computer where [[it]] the encrypted intermediate results set is decrypted and operated upon by the client computer to create an updated intermediate results set, and the updated intermediate results set is then re-encrypted and returned to the server computer where [[it]] the re-encrypted intermediate results set is further operated upon by the server computer ~~before being~~ to generate an encrypted new intermediate results set, which is sent ~~again~~ from the server computer to the client computer to be decrypted and operated upon by the client computer in order to produce actual results for presentation to a user.

17. (CURRENTLY AMENDED) A method of performing computations on encrypted data stored on a computer system, comprising:

encrypting data at a client computer;

hosting the encrypted data on a server computer;

operating upon the encrypted data at the server computer to produce an intermediate results set;

transferring the intermediate results set from the server computer to the client computer;

decrypting the transferred intermediate results set at the client computer;

operating upon the ~~transferred~~ decrypted intermediate results set at the client computer to generate an updated intermediate results set;

re-encrypting the updated intermediate results set at the client computer;

transferring the re-encrypted intermediate results set to the server computer;

operating upon the ~~transferred~~ re-encrypted intermediate results set at the server computer to generate a new intermediate results set;

transferring the new intermediate results set from the server computer to the client computer; [[and]]

decrypting the transferred new intermediate results set at the client computer; and

producing actual results from the ~~transferred~~ decrypted new intermediate results set at the client computer for presentation to a user.

18. (CURRENTLY AMENDED) The method of claim 17, wherein the client computer decrypts the intermediate results set, performs one or more operations on the decrypted intermediate results set to generate [[an]] the updated intermediate results set, re-encrypts the updated intermediate results set, and returns the re-encrypted intermediate results set to the server computer.

19. (ORIGINAL) The method of claim 18, wherein the operations comprise logical comparison operations.

20. (ORIGINAL) The method of claim 18, wherein the operations comprise filtering operations.

21. (ORIGINAL) The method of claim 18, wherein the operations comprise sorting operations.

22. (ORIGINAL) The method of claim 17, wherein the server computer executes a round-trip filtering operator that specifies when the intermediate results set is sent from the server computer to the client computer to be operated upon by the client computer and then returned to the server computer to be operated upon by the server computer.

23. (ORIGINAL) The method of claim 22, wherein the round-trip filtering operator sends maybe tuples from the server computer to the client computer, the client computer filters out certain tuples from the maybe tuples, and the server computer receives back only certain tuples from the client computer.

24. (CURRENTLY AMENDED) The method of claim 22, wherein the server receives a query from the client computer, generates a plurality of query execution plans having different placements of the round-trip filtering operator, and chooses one of query execution plans that ~~optimizes~~ comprises an optimal placement of the round-trip filtering operator in a query tree.

25. (ORIGINAL) The method of claim 22, wherein the server executes a last-trip decryption operator that specifies when the intermediate results set is sent from the server computer to the client computer in order to produce actual results.

26. (CURRENTLY AMENDED) The method of claim 25, wherein the server computer receives a query from the client computer, generates a plurality of query execution plans having different placements of the last-trip decryption operator, and chooses one of query execution plans that ~~optimizes~~ comprises an optimal placement of the last-trip decryption operator in a query tree.

27. (CURRENTLY AMENDED) The method of claim 26, wherein the server computer chooses one of the query execution plans that ~~optimizes~~ comprises an optimal placement for the round-trip filtering operators in the query tree and then chooses one of the query execution plans that ~~optimizes~~ comprises an optimal placement for the last-trip decryption operator in the query tree.

28. (ORIGINAL) The method of claim 25, wherein the last-trip decryption operator can be pulled up above any unary and binary operator in a query tree, except for a GroupBy operator.

29. (ORIGINAL) The method of claim 28, wherein the GroupBy operator can be pulled above a unary operator other than another GroupBy operator if and only if all columns used in the unary operator are functionally computed by grouping columns of an input relation.

30. (CURRENTLY AMENDED) The method of claim 29, wherein the GroupBy operator can be pulled above a binary operator if:

[[(1)]] a left-side relation of the GroupBy operator has a key; and

[[(2)]] a predicate of the GroupBy operator does not use a result of the GroupBy operator.

31. (CURRENTLY AMENDED) An article of manufacture ~~embodying logic for~~ comprising a storage device for storing instructions that, when read and executed by one or more

computers, result in the computers performing computations on encrypted data stored on a computer system, ~~the logic~~ comprising:

encrypting data at a client computer;

hosting the encrypted data on a server computer;

operating upon the encrypted data at the server computer to produce an intermediate results set;

transferring the intermediate results set from the server computer to the client computer;

decrypting the transferred intermediate results set at the client computer;

operating upon the ~~transferred~~ decrypted intermediate results set at the client computer to generate an updated intermediate results set;

re-encrypting the updated intermediate results set at the client computer;

transferring the re-encrypted intermediate results set to the server computer;

operating upon the ~~transferred~~ re-encrypted intermediate results set at the server computer to generate a new intermediate results set;

transferring the new intermediate results set from the server computer to the client computer; [[and]]

decrypting the transferred new intermediate results set at the client computer; and

producing actual results from the ~~transferred~~ decrypted new intermediate results set at the client computer for presentation to a user.

32. (CURRENTLY AMENDED) The article of claim 31, wherein the client computer decrypts the intermediate results set, performs one or more operations on the decrypted intermediate results set to generate [[an]] the updated intermediate results set, re-encrypts the updated intermediate results set, and returns the re-encrypted intermediate results set to the server computer.

33. (ORIGINAL) The article of claim 32, wherein the operations comprise logical comparison operations.

34. (ORIGINAL) The article of claim 32, wherein the operations comprise filtering operations.

35. (ORIGINAL) The article of claim 32, wherein the operations comprise sorting operations.

36. (ORIGINAL) The article of claim 31, wherein the server computer executes a round-trip filtering operator that specifies when the intermediate results set is sent from the server computer to the client computer to be operated upon by the client computer and then returned to the server computer to be operated upon by the server computer.

37. (ORIGINAL) The article of claim 36, wherein the round-trip filtering operator sends maybe tuples from the server computer to the client computer, the client computer filters out certain tuples from the maybe tuples, and the server computer receives back only certain tuples from the client computer.

38. (CURRENTLY AMENDED) The article of claim 36, wherein the server receives a query from the client computer, generates a plurality of query execution plans having different placements of the round-trip filtering operator, and chooses one of query execution plans that ~~optimizes~~ comprises an optimal placement of the round-trip filtering operator in a query tree.

39. (ORIGINAL) The article of claim 36, wherein the server executes a last-trip decryption operator that specifies when the intermediate results set is sent from the server computer to the client computer in order to produce actual results.

40. (CURRENTLY AMENDED) The article of claim 39, wherein the server computer receives a query from the client computer, generates a plurality of query execution plans having different placements of the last-trip decryption operator, and chooses one of the query execution plans that ~~optimizes~~ comprises an optimal placement of the last-trip decryption operator in a query tree.

41. (CURRENTLY AMENDED) The article of claim 40, wherein the server computer chooses one of the query execution plans that ~~optimizes~~ comprises an optimal placement for the

round-trip filtering operators in the query tree and then chooses one of the query execution plans that ~~optimizes~~ comprises an optimal placement for the last-trip decryption operator in the query tree.

42. (ORIGINAL) The article of claim 39, wherein the last-trip decryption operator can be pulled up above any unary and binary operator in a query tree, except for a GroupBy operator.

43. (ORIGINAL) The article of claim 42, wherein the GroupBy operator can be pulled above a unary operator other than another GroupBy operator if and only if all columns used in the unary operator are functionally computed by grouping columns of an input relation.

44. (CURRENTLY AMENDED) The article of claim 43, wherein the GroupBy operator can be pulled above a binary operator if:

[[(1)]] a left-side relation of the GroupBy operator has a key; and

[[(2)]] a predicate of the GroupBy operator does not use a result of the GroupBy operator.